On the Arithmetic Complexity of Strassen-Like Matrix Multiplications

Murat Cenk and M. Anwar Hasan

Abstract

The Strassen algorithm for multiplying 2×2 matrices requires seven multiplications and 18 additions. The recursive use of this algorithm for matrices of dimension n yields a total arithmetic complexity of $(7n^{2.81} - 6n^2)$ for $n = 2^k$. Winograd showed that using seven multiplications for this kind of multiplications is optimal, so any algorithm for multiplying 2×2 matrices with seven multiplications is therefore called a Strassen-like algorithm. Winograd also discovered an additively optimal Strassen-like algorithm with 15 additions. This algorithm is called the Winograd's variant, whose arithmetic complexity is $(6n^{2.81} - 5n^2)$ for $n = 2^k$ and $(3.73n^{2.81} - 5n^2)$ for $n = 8 \cdot 2^k$, which is the best-known bound for Strassen-like multiplications. This paper proposes a method that reduces the complexity of Winograd's variant to $(5n^{2.81} + 0.5n^{2.59} + 2n^{2.32} - 6.5n^2)$ for $n = 2^k$. It is also shown that the total arithmetic complexity can be improved to $(3.55n^{2.81} + 0.148n^{2.59} + 1.02n^{2.32} - 6.5n^2)$ for $n = 8 \cdot 2^k$, which, to the best of our knowledge, improves the best-known bound for a Strassen-like matrix multiplication algorithm.

Index Terms

Fast matrix multiplication, Strassen-like matrix multiplication, computational complexity, cryptographic computations, computer algebra.

1 INTRODUCTION

Let $O(n^\omega)$ be the complexity of multiplying two $n \times n$ matrices. An ordinary matrix multiplication algorithm requires n^3 multiplications and $(n^3 - n^2)$ additions, which means that, $\omega \leq 3$ for the ordinary method. In 1969, Strassen [15] showed that two 2×2 matrices can be multiplied with seven multiplications rather than eight. The recursive use of this algorithm yields $\omega \leq 2.81$. In 1978 and 1980, Pan [9], [10], [11] used his trilinear aggregating techniques to obtain $\omega \leq 2.795$ and $\omega \leq 2.781$, respectively. In other work, in 1979, Bini et al. [1] presented approximation algorithms and produced one with $\omega \leq 2.7799$. Schönhage [13] introduced the concept of disjoint matrix multiplication in 1981 and was able to obtain $\omega \leq 2.5479$. In

• *Department of Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1* {mcenk,ahasan@uwaterloo.ca}

1986, Strassen [16] obtained $\omega \leq 2.4785$ by introducing a new method called the laser method, which was used by Coppersmith and Winograd [4] in 1987 in order to determine the well known bound $\omega \leq 2.376$. This upper bound has recently been reduced to $w \leq 2.374$ by Stothers [14] and to $w \leq 2.373$ by Williams [18] through the use of constructions similar to those of Coppersmith and Winograd. On the other hand, in 2003, Cohn and Umans [3] approached this problem by introducing a new group-theoretic approach. Cohn et al. [2] also proposed several multiplication algorithms using this approach, but the bounds were no better than the Coppersmith-Winograd's results.

One of the algorithms most widely employed for practical applications is the algorithm that uses seven multiplications for multiplying 2×2 matrices, as proposed by Strassen [15] in 1969. Winograd [19] proved that number of multiplications is optimal, so the algorithms using seven multiplications for 2×2 matrix multiplications are thus called Strassen-like algorithms. In [12], it was shown that the optimal number of additions in a Strassen-like algorithm is 15, and Winograd proposed such an algorithm that uses seven multiplications and 15 additions. This algorithm is called Winograd's variant. Strassen-like algorithms provide grater efficiency for sizes in practical use than other algorithms that have better matrix exponent because of the hidden factor in big-O notation. It should be noted that Pan's trilinear aggregation techniques [9], [10] also yield practical algorithms. For example, Kaporin [8] worked on Pan's techniques and compared their complexities with that of the Winograd's variant. He reported that Pan's techniques yield an arithmetic complexity of $(4.894n^{2.7760} - 16.16n^2)$ for $n = 18 \cdot 48^k$ and that this complexity provides a computational time comparable to that produced by Strassen-like algorithms for matrices of medium-large size, $2000 \leq n \leq 10000$.

The work presented in this paper deals with the arithmetic complexity of widely used Strassen-like algorithms such as ones found in cryptographic computations [7], [17], in which the matrices are generally over finite fields and no stability problems exist. For this study, the-best known Strassen-like arithmetic complexities have been decreased from $(6n^{2.81} - 5n^2)$ to $(5n^{2.81} + 0.5n^{2.59} + 2n^{2.32} - 6.5n^2)$ for $n = 2^k$ and from $(3.73n^{2.81} - 5n^2)$ to $(3.55n^{2.81} + 0.148n^{2.59} + 1.02n^{2.32} - 6.5n^2)$ for $n = 8 \cdot 2^k$, i.e., when the algorithm is stopped at the point when the size of matrices becomes eight and then the ordinary method is applied.

Notation and model of computation: The matrices that appear throughout the paper are over an arbitrary ring $\mathcal{R}$. The dimension of matrices is shown by n and $n = 2^k$ is assumed for a positive integer k. $M_{\otimes}(n)$ and $M_{\oplus}(n)$ denote the number of multiplications and additions/subtractions in $\mathcal{R}$ needed for multiplying $n \times n$ matrices over $\mathcal{R}$, respectively. The total arithmetic cost, i.e. the sum of multiplications and additions/subtractions is denoted by $M(n)$. SA and WV represent the Strassen algorithm and Winograd's variant of the Strassan algorithm, respectively. The operations $\oplus, \ominus$ and $\otimes$ are used for componentwise vector addition, subtraction and multiplication, respectively. The other notations employed in this paper are CMF, CM, CA and R, which represent component matrix formation, component multiplication, component addition and reconstruction, respectively. Let X be any of CMF, CM, CA or R. $M_{\otimes}^{X}(n)$ and $M_{\oplus}^{X}(n)$ denote the number of multiplications and additions/subtractions in $\mathcal{R}$ needed for computing

the X, respectively. In the work presented in this paper, the arithmetic complexity of the algorithms is computed for the multiplication of matrices over an arbitrary ring $\mathcal{R}$, i.e. we compute the number of multiplications and additions/subtractions in $\mathcal{R}$ required for multiplying two matrices. Other problems, such as memory usage or the numerical stability of matrix multiplications, are beyond the scope of this work.

The remainder of the paper is organized as follows: The algorithms SA and WV are introduced in the next section, followed by the presentation of the block decomposition of SA and WV in section 3. The proposed improved complexities of WV are explained in section 4 and an analysis of the complexities obtained by stopping the recursion early is provided in section 5. Section 6 includes a discussion of further improvements using a block recombination method and the final two sections of the paper provide a comparison of all of the complexities as well as conclusions that can be drawn.

2 Matrix Multiplication

This section introduces the algorithms SA and WV, together with their arithmetic complexities. For all of the work presented in this paper, the following theorem is useful for solving the recursive equations of the algorithms as a means of determining the asymptotical bounds. Its proof can be found in [5].

Theorem 1. *[5] (Master theorem) Let $a \geq 1$ and $b > 1$ be constants, $f(n)$ be a function, and $M(n)$ be defined on nonnegative integers by the recurrence*

$$M(n) = aM(n/b) + f(n),$$

where if n is not divisible by b, use $\lceil n/b \rceil$. Then $M(n)$ can be bounded asymptotically as follows:

1) *If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $M(n) = \Theta(n^{\log_b a})$.*
2) *If $f(n) = \Theta(n^{\log_b a})$, then $M(n) = \Theta(n^{\log_b a} \log_b(n))$*
3) *If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $M(n) = \Theta(f(n))$.*

Strassen algorithm (SA): The ordinary matrix multiplication method for two $n \times n$ matrices requires $O(n^3)$ operations, more specifically n^3 multiplications and $(n^3 - n^2)$ additions. In [15], Strassen proposed an algorithm for multiplying matrices faster than with the ordinary algorithm. In SA, two 2×2 matrices are multiplied with seven multiplications and 18 additions. The recursive use of this algorithm reduces the arithmetic complexity to $O(n^{\log_2 7})$. The explicit algorithm is as follows: Let A and B be matrices of size $n = 2^k$ for a positive integer k, and $C = AB$ be their product. These matrices can be written as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \; B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \; C = AB = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}, \tag{1}$$

where A_{ij}, B_{ij} and C_{ij} for $1 \leq i, j \leq 2$ are $2^{k-1} \times 2^{k-1}$ matrices. SA is the following:

$$\begin{cases} P_1 = (A_{11} + A_{22})(B_{11} + B_{22}),\ P_2 = (A_{21} + A_{22})B_{11},\ P_3 = A_{11}(B_{12} - B_{22}),\ P_4 = A_{22}(-B_{11} + B_{21}), \\ P_5 = (A_{11} + A_{12})B_{22},\ P_6 = (-A_{11} + A_{21})(B_{11} + B_{12}),\ P_7 = (A_{12} - A_{22})(B_{21} + B_{22}) \\ C_{11} = P_1 + P_4 - P_5 + P_7,\ C_{12} = P_3 + P_5,\ C_{21} = P_2 + P_4,\ C_{22} = P_1 - P_2 + P_3 + P_6. \end{cases} \tag{2}$$

Based on Theorem 1, the complexities of SA are as follows:

$$\begin{cases} M_{\otimes}(n) \leq 7M_{\otimes}(\frac{n}{2}),\ M_{\otimes}(1) = 1 \implies M_{\otimes}(n) = n^{\log_2 7}, \\ M_{\oplus}(n) \leq 7M_{\oplus}(\frac{n}{2}) + 18(\frac{n}{2})^2,\ M_{\oplus}(1) = 0, \implies M_{\oplus}(n) = 6n^{\log_2 7} - 6n^2, \\ M(n) \leq 7M(\frac{n}{2}) + 18(\frac{n}{2})^2,\ M(1) = 1, \implies M_{\oplus}(n) = 7n^{\log_2 7} - 6n^2. \end{cases} \tag{3}$$

Winograd's variant (WV): WV uses seven multiplications and 15 additions for multiplying 2×2 matrices. Let A, B and C be as in (1). WV is then the following:

$$\begin{cases} P_1 = A_{11}B_{11},\ P_2 = A_{12}B_{21},\ P_3 = A_{22}(B_{11} - B_{12} - B_{21} + B_{22}), \\ P_4 = (A_{11} - A_{21})(-B_{12} + B_{22}),\ P_5 = (A_{21} + A_{22})(-B_{11} + B_{12}), \\ P_6 = (A_{11} + A_{12} - A_{21} - A_{22})B_{22},\ P_7 = (A_{11} - A_{21} - A_{22})(B_{11} - B_{12} + B_{22}), \\ C_{11} = P_1 + P_2,\ C_{12} = P_1 + P_5 + P_6 - P_7,\ C_{21} = P_1 - P_3 + P_4 - P_7,\ C_{22} = P_1 + P_4 + P_5 - P_7. \end{cases} \tag{4}$$

It should be noted that $(A_{11} - A_{21})$ in P_4 is also used in P_7 and $(A_{11} - A_{21} - A_{22})$ in P_7 is also used in P_6. Similarly, $(-B_{12} + B_{22})$ in P_4 is also used in P_7 and $(B_{11} - B_{12} + B_{22})$ in P_7 is also used in P_3. Eight additions are therefore needed for computing P_i's. On the other hand, $(P_1 - P_7)$ is a common sum in C_{12}, C_{21}, and C_{22}, and $(P_1 + P_5 - P_7)$ is a common sum in C_{12} and C_{22}. Seven additions are required for the computations of each of C_{ij}. As a result, based on Theorem 1, the complexities of WV can be computed as follows:

$$\begin{cases} M_{\otimes}(n) \leq 7M_{\otimes}(\frac{n}{2}),\ M_{\otimes}(1) = 1, \implies M_{\otimes}(n) = n^{\log_2 7} \\ M_{\oplus}(n) \leq 7M_{\oplus}(\frac{n}{2}) + 15(\frac{n}{2})^2,\ M_{\oplus}(1) = 0, \implies M_{\oplus}(n) = 5n^{\log_2 7} - 5n^2, \\ M(n) \leq 7M(\frac{n}{2}) + 15(\frac{n}{2})^2,\ M(1) = 1, \implies M(n) = 6n^{\log_2 7} - 5n^2. \end{cases} \tag{5}$$

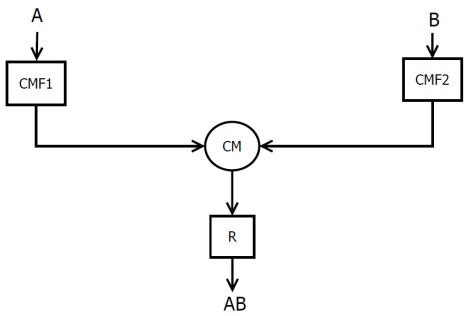
3 BLOCK DECOMPOSITION OF MATRIX MULTIPLICATION

To demonstrate the use of SA and WV recursively, this section describes the decomposition of SA and WV into three main blocks as shown in Figure 1: component matrix formation (CMF), component multiplication (CM) and reconstruction (R) [6]. To multiply matrices A and B of sizes $n \times n$, the first step is to compute all of the linear combinations of A_{ij}'s and B_{ij}'s for $1 \leq i, j \leq 2$, which correspond to the left hand and right hand factors of the multiplications in SA or WV. This step is called CMF (Figure 1), and the CMF which applied to A is called CMF_1, and the CMF which is applied to B is called CMF_2. The size of CMFs is $n^{\log_2 7} = 7^k$, because the CMF entries are split into seven parts in each recursion. Those linear combinations are then multiplied componentwise in order to construct the products $P_1, \ldots, P_7$; this step is called CM. Since $CMF_1(A)$ and $CMF_2(B)$ are multiplied component by component, the size of

this step is $n^{\log_2 7} = 7^k$. Finally, linear combinations of these products are computed in order to obtain the result, which is R, with a size of $n^2 = 4^k$. The following sections include details of these blocks for SA and WV.

Remark 1. It should be noted that the operations $\oplus, \ominus$ and $\otimes$ are used for componentwise vector addition, subtraction and multiplication, respectively. For example, $CMF_1(A_1) \oplus CMF_1(A_2)$ represents the componentwise addition of vectors $CMF_1(A_1)$ and $CMF_1(A_2)$ for matrices A_1 and A_2.

Fig. 1. Data flow of matrix multiplication with block decomposition



3.1 Block decomposition of the Strassen algorithm

The three blocks of SA and their complexities are given below.

3.1.1 Component matrix formation (CMF).

For an $n \times n$ matrix A, CMF_1 is defined for SA as follows:

$$\begin{cases} R_1 = A_{11} + A_{22},\ R_2 = A_{21} + A_{22},\ R_3 = A_{11} + A_{12},\ R_4 = -A_{11} + A_{21},\ R_5 = A_{12} - A_{22}. \\ CMF_1(A) = A_{11} \text{ for } n = 1, \\ CMF_1(A) = (CMF_1(R_1), CMF_1(R_2), CMF_1(A_{11}), CMF_1(A_{22}), CMF_1(R_3), CMF_1(R_4), \\ CMF_1(R_5)) \text{ for } n \geq 2, \end{cases} \tag{6}$$

For B, it is defined as

$$\begin{cases} R_6 = B_{11} + B_{22},\ R_7 = B_{12} - B_{22},\ R_8 = -B_{11} + B_{21},\ R_9 = B_{11} + B_{12},\ R_{10} = B_{21} + B_{22}. \\ CMF_2(B) = B_{11} \text{ for } n = 1, \\ CMF_2(B) = (CMF_2(R_6), CMF_2(B_{11}), CMF_2(R_7), CMF_2(R_8), CMF_2(B_{22}), CMF_2(R_9), \\ CMF_2(R_{10})) \text{ for } n \geq 2. \end{cases} \tag{7}$$

It should be noted that the sizes of $CMF_1(A)$ and $CMF_2(B)$ are $n^{\log_2 7} = 7^k$ each and that their complexities are identical, requiring seven CMFs which applied to $n/2 \times n/2$ matrices plus five additions of $n/2 \times n/2$ matrices. The CMFs of SA therefore has the following complexity:

$$M_{\oplus}^{CMF}(n) \leq 7M_{\oplus}^{CMF}(n/2) + 5(n/2)^2,\ \ M_{\oplus}^{CMF}(1) = 0 \Longrightarrow M_{\oplus}^{CMF}(n) = (5/3)n^{\log_2 7} - (5/3)n^2.$$

3.1.2 Component Multiplication (CM).

For CM, two vectors of dimension $n^{\log_2 7} = 7^k$ are multiplied component by component so that the size of it is $n^{\log_2 7} = 7^k$ and we have

$$M_{\otimes}^{CM}(n) \leq 7M_{\otimes}^{CM}(\frac{n}{2}), \;\; M_{\otimes}^{CM}(1) = 1 \Longrightarrow M_{\otimes}^{CM}(n) = n^{\log_2 7}.$$

3.1.3 Reconstruction (R).

Let C be a vector of length $n^{\log_2 7}$. Assume that $C = (C_1)$ for $n = 1$ where the length of C_1 is one , and $C = (C_1, C_2, \ldots, C_7)$ for $n \geq 2$ where the lengths of C_i's for $i = 1, \ldots, 7$ are $n^{\log_2 7}/7$. The reconstruction $R(C)$ is then computed recursively as follows:

$$\begin{cases} R(C) = C_1 \text{ for } n = 1, \\ R(C) = (R(C_1) \oplus R(C_4) \ominus R(C_5) \oplus R(C_7), R(C_3) \oplus R(C_5), R(C_2) \oplus R(C_4), \\ R(C_1) \ominus R(C_2) \oplus R(C_3) \oplus R(C_6)) \text{ for } n \geq 2. \end{cases} \tag{8}$$

It should be noted that the size of $R(C)$ is $n^2 = 4^k$, and the complexity of this block is

$$M_{\oplus}^{R}(n) \leq 7M_{\oplus}^{R}(n/2) + 8(n/2)^2, \;\; M_{\oplus}^{R}(1) = 0 \Longrightarrow M_{\oplus}^{R}(n) = (8/3)n^{\log_2 7} - (8/3)n^2.$$

The complexities of the different sub-blocks of SA are listed in Table 1. As can be seen clearly in Figure 1, the complexity of SA requires $Q_1 = CMF_1(A)$, $Q_2 = CMF_2(B)$, $Q_3 = CM(Q_1, Q_2)$ and $Q_4 = R(Q_3)$. The complexity of SA is therefore computed using the complexities of those blocks given in Table 1, as follows:

$$2M_{\oplus}^{CMF}(n) + M_{\otimes}^{CM}(n) + M_{\oplus}^{R}(n) = 7n^{\log_2 7} - 6n^2.$$

The CMFs, CM and R of SA for $n = 2$ are shown in the following example:

Example 1. Consider the case of $n = 2$. Let

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \; B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}.$$

The CMFs, CM and R for the Strassen algorithm are then the followings:

$$CMF_1(A) = (a_{11} + a_{22}, a_{21} + a_{22}, a_{11}, a_{22}, a_{11} + a_{12}, -a_{11} + a_{21}, a_{12} - a_{22}),$$
$$CMF_2(B) = (b_{11} + b_{22}, b_{11}, b_{12} - b_{22}, -b_{11} + b_{21}, b_{22}, b_{11} + b_{12}, b_{21} + b_{22}).$$

On the other hand, CM of $CMF_1(A)$ and $CMF_2(B)$ are as follows:

$$CM(CMF_1(A), CMF_2(B)) = ((a_{11} + a_{22})(b_{11} + b_{22}), \ldots, (a_{12} - a_{22})(b_{21} + b_{22})) = (P_1, P_2, \ldots, P_7) = P,$$

where P_i's are the same with in (2). Finally the reconstruction block is given by

$$R(P) = (P_1 + P_4 - P_5 + P_7, P_3 + P_5, P_2 + P_4, P_1 - P_2 + P_3 + P_6).$$

3.2 Block decomposition of Winograd's variant

The three blocks of Winograd's variant and their complexities are presented below.

3.2.1 Component matrix formation (CMF).

For A, define

$$\begin{cases} R_1 = A_{11} - A_{21},\ R_2 = A_{21} + A_{22},\ R_3 = R_1 - A_{22},\ R_4 = R_3 + A_{12} \\ CMF_1(A) = A_{11} \text{ for } n = 1, \\ CMF_1(A) = (CMF_1(A_{11}), CMF_1(A_{12}), CMF_1(A_{22}), CMF_1(R_1), CMF_1(R_2), CMF_1(R_4), \\ CMF_1(R_3)), \text{ for } n \geq 2, \end{cases} \tag{9}$$

and for B, define

$$\begin{cases} R_5 = -B_{12} + B_{22},\ R_6 = -B_{11} + B_{12},\ R_7 = -R_6 + B_{22},\ R_8 = R_7 - B_{21} \\ CMF_2(B) = B_{11} \text{ for } n = 1, \\ CMF_2(B) = (CMF_2(B_{11}), CMF_2(B_{21}), CMF_2(R_8), CMF_2(R_5), CMF_2(R_6), CMF_2(B_{22}), \\ CMF_2(R_7)) \text{ for } n \geq 2. \end{cases} \tag{10}$$

The complexity of these operations is identical:

$$M_{\oplus}^{CMF}(n) \leq 7M_{\oplus}^{CMF}(n/2) + 4(n/2)^2,\ \ M_{\oplus}^{CMF}(1) = 0 \Longrightarrow M_{\oplus}^{CMF}(n) = (4/3)n^{\log_2 7} - (4/3)n^2.$$

Example 2. This example is an explicit demonstration of the CMF_1 operation for $n = 4$. To save space, only the CMF_1 operation is presented. Let four sub-matrices of dimension 2×2 be constructed as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

$$A_{11} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix},\ A_{12} = \begin{bmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \end{bmatrix},\ A_{21} = \begin{bmatrix} a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix},\ A_{22} = \begin{bmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{bmatrix}.$$

The original CMF_1 of A is now computed. From (9), we find that:

$$CMF_1(A) = (CMF_1(A_{11}), CMF_1(A_{12}), CMF_1(A_{22}), CMF_1(R_1), CMF_1(R_2), CMF_1(R_4), CMF_1(R_3)),$$

where $R_1 = A_{11} - A_{21}$, $R_2 = A_{21} + A_{22}$, $R_3 = R_1 - A_{22}$, and $R_4 = R_3 + A_{12}$. Therefore,

$$R_1 = \begin{bmatrix} \underbrace{(a_{11} - a_{31})}_{s_1} & \underbrace{(a_{12} - a_{32})}_{s_2} \\ \underbrace{(a_{21} - a_{41})}_{s_3} & \underbrace{(a_{22} - a_{42})}_{s_4} \end{bmatrix},\ R_3 = \begin{bmatrix} \underbrace{(s_1 - a_{33})}_{s_9} & \underbrace{(s_2 - a_{34})}_{s_{10}} \\ \underbrace{(s_3 - a_{43})}_{s_{11}} & \underbrace{(s_4 - a_{44})}_{s_{12}} \end{bmatrix},$$

$$R_2 = \begin{bmatrix} \underbrace{(a_{31}+a_{33})}_{s_5} & \underbrace{(a_{32}+a_{34})}_{s_6} \\ \underbrace{(a_{41}+a_{43})}_{s_7} & \underbrace{(a_{42}+a_{44})}_{s_8} \end{bmatrix}, R_4 = \begin{bmatrix} \underbrace{(s_9+a_{13})}_{s_{13}} & \underbrace{(s_{10}+a_{14})}_{s_{14}} \\ \underbrace{(s_{11}+a_{23})}_{s_{15}} & \underbrace{s_{12}+a_{24})}_{s_{16}} \end{bmatrix}.$$

It should be noted that the cost of computing R_i's for $i = 1, \ldots, 4$ is 16 additions. On the other hand, the computation of CMFs applied to 2×2 matrices are the following:

$$\begin{aligned}
CMF_1(A_{11}) = \quad & (a_{11}, a_{12}, a_{22}, \underbrace{a_{11}-a_{21}}_{r_1}, \underbrace{a_{21}+a_{22}}_{r_2}, \underbrace{r_3+a_{12}}_{r_4}, \underbrace{r_1-a_{22}}_{r_3}), \\
CMF_1(A_{12}) = \quad & (a_{13}, a_{14}, a_{24}, \underbrace{a_{13}-a_{23}}_{r_5}, \underbrace{a_{23}+a_{24}}_{r_6}, \underbrace{r_7+a_{14}}_{r_8}, \underbrace{r_5-a_{24}}_{r_7}), \\
CMF_1(A_{22}) = \quad & (a_{33}, a_{34}, a_{44}, \underbrace{a_{33}-a_{43}}_{r_9}, \underbrace{a_{43}+a_{44}}_{r_{10}}, \underbrace{r_{11}+a_{34}}_{r_{12}}, \underbrace{r_9-a_{44}}_{r_{11}}], \\
CMF_1(R_1) = \quad & (s_1, s_2, s_4, \underbrace{s_1-s_3}_{r_{13}}, \underbrace{s_3+s_4}_{r_{14}}, \underbrace{r_{15}+s_2}_{r_{16}}, \underbrace{r_{13}-s_4}_{r_{15}}), \\
CMF_1(R_2) = \quad & (s_5, s_6, s_8, \underbrace{s_5-s_7}_{r_{17}}, \underbrace{s_7+s_8}_{r_{18}}, \underbrace{r_{19}+s_6}_{r_{20}}, \underbrace{r_{17}-s_8}_{r_{19}}), \\
CMF_1(R_4) = \quad & (s_{13}, s_{14}, s_{16}, \underbrace{s_{13}-s_{15}}_{r_{21}}, \underbrace{s_{15}+s_{16}}_{r_{22}}, \underbrace{r_{23}+s_{14}}_{r_{24}}, \underbrace{r_{21}-s_{16}}_{r_{23}}), \\
CMF_1(R_3) = \quad & (s_9, s_{10}, s_{12}, \underbrace{s_9-s_{11}}_{r_{25}}, \underbrace{s_{11}+s_{12}}_{r_{26}}, \underbrace{r_{27}+s_{10}}_{r_{28}}, \underbrace{r_{25}-s_{12}}_{r_{27}}).
\end{aligned}$$

It should be noted that 28 additions/subtractions are needed for the computation of r_i's where $i = 1, \ldots, 28$. The computation of the original $CMF_1(A)$ thus requires a total of 44 additions/subtractions.

3.2.2 *Component Multiplication (CM).*

After $CMF(A)$ and $CMF(B)$ are computed, they are multiplied component by component, resulting in

$$M_{\otimes}^{CM}(n) \leq 7M_{\otimes}^{CM}(\frac{n}{2}), \;\; M_{\otimes}^{CM}(1) = 1 \Longrightarrow M_{\otimes}^{CM}(n) = n^{\log_2 7}.$$

3.2.3 *Reconstruction (R).*

Let C be as in section 3.1.3. Following the component multiplication, the reconstruction $R(C)$ is computed recursively, as follows:

$$\begin{cases}
R(C) = C_1 \text{ for } n = 1, \\
R(C) = (R(C_1) \oplus R(C_2), \underbrace{S_1 \oplus R(C_5)}_{S_2} \oplus R(C_6), \underbrace{R(C_1) \ominus R(C_7)}_{S_1} \ominus R(C_3) \oplus R(C_4), \\
S_2 \oplus R(C_4)) \text{ for } n \geq 2.
\end{cases} \tag{11}$$

The complexity of this block is

$$M_{\oplus}^{R}(n) \leq 7M_{\oplus}^{R}(\frac{n}{2}) + 7(\frac{n}{2})^2, \;\; M_{\oplus}^{R}(1) = 0 \Longrightarrow M_{\oplus}^{R}(n) = (7/3)n^{\log_2 7} - (7/3)n^2.$$

Example 3. Consider the case $n = 4$. The length of CM is 49, which is the input of R. Assume that $C = (C_1, \ldots, C_7)$ where $C_i = (P_{7i-6}, \ldots P_{7i})$ for $i = 1, \ldots, 7$. The first step is to compute the $R(C_i)$s, following which the result is then obtained using (11):

$$\begin{aligned}
R(C_1) = \quad & (P_1 + P_2, \underbrace{r_1 + P_5}_{r_2} + P_6, \underbrace{P_1 - P_7}_{r_1} - P_3 + P_4, r_2 + P_4),\\
R(C_2) = \quad & (P_8 + P_9, \underbrace{r_3 + P_{12}}_{r_4} + P_{13}, \underbrace{P_8 - P_{14}}_{r_3} - P_{10} + P_{11}, r_4 + P_{11}),\\
R(C_3) = \quad & (P_{15} + P_{16}, \underbrace{r_5 + P_{19}}_{r_6} + P_{20}, \underbrace{P_{15} - P_{21}}_{r_5} - P_{17} + P_{18}, r_6 + P_{18}),\\
R(C_4) = \quad & (P_{22} + P_{23}, \underbrace{r_7 + P_{26}}_{r_8} + P_{27}, \underbrace{P_{22} - P_{28}}_{r_7} - P_{24} + P_{25}, r_8 + P_{25}),\\
R(C_5) = \quad & (P_{29} + P_{30}, \underbrace{r_9 + P_{33}}_{r_{10}} + P_{34}, \underbrace{P_{29} - P_{35}}_{r_9} - P_{31} + P_{32}, r_{10} + P_{32}),\\
R(C_6) = \quad & (P_{36} + P_{37}, \underbrace{r_{11} + P_{40}}_{r_{12}} + P_{41}, \underbrace{P_{36} - P_{42}}_{r_{11}} - P_{38} + P_{39}, r_{12} + P_{39}),\\
R(C_7) = \quad & (P_{43} + P_{44}, \underbrace{r_{13} + P_{47}}_{r_{14}} + P_{48}, \underbrace{P_{43} - P_{49}}_{r_{13}} - P_{45} + P_{46}, r_{14} + P_{46}).
\end{aligned}$$

Since each $R(C_i)$ requires seven additions, the computation of all $R(C_i)$'s requires 49 additions/subtractions, with the following result:

$$R(C) = (R(C_1) \oplus R(C_2), \underbrace{S_1 \oplus R(C_5)}_{S_2} \oplus R(C_6), \underbrace{R(C_1) \ominus R(C_7)}_{S_1} \ominus R(C_3) \oplus R(C_4), S_2 \oplus R(C_4)),$$

which requires 28 additions because each $(R(C_i) \oplus R(C_j))$ or $(R(C_i) \ominus R(C_j))$ for $i, j \in \{1, \ldots, 7\}$ needs four additions. As a result, R for $n = 4$ requires a total of 77 additions.

The complexity of WV requires $Q_1 = CMF_1(A)$, $Q_2 = CMF_2(B)$, $Q_3 = CM(Q_1, Q_2)$, and $Q_4 = R(Q_3)$ so that

$$2M_{\oplus}^{CMF}(n) + M_{\otimes}^{CM}(n) + M_{\oplus}^{R}(n) = 6n^{\log_2 7} - 5n^2$$

based on the complexities of these blocks as listed in Table 1.

TABLE 1
Complexities of the different sub-operations of algorithms

Method	Operation	Recursion	Non-recursion
Strassen	CMF	$7M_{\oplus}^{CMF}(n/2) + 5(n/2)^2$	$1.67n^{\log_2 7} - 1.67n^2$
	CM	$7M_{\otimes}^{CM}(n/2)$	$n^{\log_2 7}$
	R	$7M_{\oplus}^{R}(n/2) + 8(n/2)^2$	$2.67n^{\log_2 7} - 2.67n^2$
Winograd	CMF	$7M_{\oplus}(n/2) + 4(n/2)^2$	$1.33n^{log_2(7)} - 1.33n^2$
	CM	$7M_{\otimes}^{CM}(\frac{n}{2})$	$n^{\log_2 7}$
	R	$7M_{\oplus}^{R}(\frac{n}{2}) + 7(\frac{n}{2})^2$	$2.33n^{\log_2 7} - 2.33n^2$

4 IMPROVED COMPLEXITIES FOR WINOGRAD'S VARIANT ALGORITHMS

This section presents improvements in the complexity of WV. Note that the techniques described in this section can also be applied to SA but since WV has a better additive complexity, the improvements are demonstrated only for WV. The primary basis of the method is the observation of the linearity of the CMF and R operations that are defined in the previous section.

4.1 Improved CMF

Consideration of the CMF operation given in (9) clearly shows that

$$CMF(A+B) = CMF(A) \oplus CMF(B).$$

This property can be proved by using induction. Based on this property, new CMFs are proposed as follows:

$$\begin{cases} R_1 = A_{11} - A_{21},\ R_2 = A_{21} + A_{22}, \\ CMF_1(A) = A_{11} \text{ for } n = 1, \\ CMF_1(A) = (CMF_1(A_{11}), CMF_1(A_{12}), CMF_1(A_{22}), CMF_1(R_1), CMF_1(R_2), \\ T_1 \oplus CMF_1(A_{12}), \underbrace{CMF_1(R_1) \ominus CMF_1(A_{22})}_{T_1}), \text{ for } n \geq 2. \end{cases} \tag{12}$$

$$\begin{cases} R_5 = -B_{12} + B_{22},\ R_6 = -B_{11} + B_{12}, \\ CMF_2(B) = B_{11} \text{ for } n = 1, \\ CMF_2(A) = (CMF_2(B_{11}), CMF_2(B_{21}), T_2 \ominus CMF_2(B_{21}), CMF_2(R_5), CMF_2(R_6), \\ CMF_2(B_{22}), \underbrace{CMF_2(B_{22}) \ominus CMF_2(R_6)}_{T_2}), \text{ for } n \geq 2. \end{cases} \tag{13}$$

It should be noted that the cost of $\oplus$ (or $\ominus$) is $(1/7)n^{\log_2 7}$ additions/subtractions because the dimension of the matrices to which CMF applied is $n/2 \times n/2$. Based on Theorem 1, the new CMF computation complexity therefore becomes

$$\begin{cases} M_{\oplus}^{CMF}(n) \leq & 5M_{\oplus}^{CMF}(n/2) + (2/7)n^{\log_2 7} + 2(n/2)^2,\ M_{\oplus}^{CMF}(1) = 0, \\ M_{\oplus}^{CMF}(n) \leq & n^{\log_2 7} + n^{\log_2 5} - 2n^2. \end{cases} \tag{14}$$

Example 4. This example explicitly shows the new CMF_1 operation for $n = 4$. For brevity, the CMF_1 operation is only presented. Let A, its sub-matrices A_{ij}'s, and R_1, R_2, $s_1, \ldots, s_8$ be as in Example 2. It should be noted that the computation of R_3 and R_4 is not required in Example 2. The next step is to compute the new CMF for A. From (12), we obtain:

$$\begin{cases} CMF_1(A) = (CMF_1(A_{11}), CMF_1(A_{12}), CMF_1(A_{22}), CMF_1(R_1), CMF_1(R_2), \\ \underbrace{Q_1 \oplus CMF_1(A_{12})}_{Q_2}, \underbrace{CMF_1(R_1) \ominus CMF_1(A_{22})}_{Q_1}). \end{cases}$$

It should be noted that the cost of computing R_1 and R_2 is 8 additions. On the other hand, the computation of CMFs applied to 2×2 matrices is as follows:

$$
\begin{aligned}
CMF_1(A_{11}) = \quad & (a_{11}, a_{12}, a_{22}, \underbrace{a_{11} - a_{21}}_{r_1}, \underbrace{a_{21} + a_{22}}_{r_2}, \underbrace{r_3 + a_{12}}_{r_4}, \underbrace{r_1 - a_{22}}_{r_3}), \\
CMF_1(A_{12}) = \quad & (a_{13}, a_{14}, a_{24}, \underbrace{a_{13} - a_{23}}_{r_5}, \underbrace{a_{23} + a_{24}}_{r_6}, \underbrace{r_7 + a_{14}}_{r_8}, \underbrace{r_5 - a_{24}}_{r_7}), \\
CMF_1(A_{22}) = \quad & (a_{33}, a_{34}, a_{44}, \underbrace{a_{33} - a_{43}}_{r_9}, \underbrace{a_{43} + a_{44}}_{r_{10}}, \underbrace{r_{11} + a_{34}}_{r_{12}}, \underbrace{r_9 - a_{44}}_{r_{11}}), \\
CMF_1(R_1) = \quad & (s_1, s_2, s_4, \underbrace{s_1 - s_3}_{r_{13}}, \underbrace{s_3 + s_4}_{r_{14}}, \underbrace{r_{15} + s_2}_{r_{16}}, \underbrace{r_{13} - s_4}_{r_{15}}), \\
CMF_1(R_2) = \quad & (s_5, s_6, s_8, \underbrace{s_5 - s_7}_{r_{17}}, \underbrace{s_7 + s_8}_{r_{18}}, \underbrace{r_{19} + s_6}_{r_{20}}, \underbrace{r_{17} - s_8}_{r_{19}}), \\
Q_1 = \quad & (\underbrace{s_1 - a_{33}}_{r_{21}}, \underbrace{s_2 - a_{34}}_{r_{22}}, \underbrace{s_4 - a_{44}}_{r_{23}}, \underbrace{r_{13} - r_9}_{r_{24}}, \underbrace{r_{14} - r_{10}}_{r_{25}}, \underbrace{r_{16} - r_{12}}_{r_{26}}, \underbrace{r_{15} - r_{11}}_{r_{27}}), \\
Q_2 = \quad & (\underbrace{r_{21} + a_{13}}_{r_{28}}, \underbrace{r_{22} + a_{14}}_{r_{29}}, \underbrace{r_{23} + a_{24}}_{r_{30}}, \underbrace{r_{24} + r_5}_{r_{31}}, \underbrace{r_{25} + r_6}_{r_{32}}, \underbrace{r_{26} + r_8}_{r_{33}}, \underbrace{r_{27} + r_7}_{r_{34}}).
\end{aligned}
$$

It should also be noted that 34 additions are needed for r_i's, $i = 1, \ldots, 34$ and eight additions are needed for s_i's for $i = 1, \ldots, 8$. The computation of the new CMF_1 therefore requires a total of 42 additions, which reduces the number of additions in the original CMF_1 computations by two.

4.2 Improved R

A new reconstruction algorithm that represents an improvement over the original one is now presented. The main idea is the following property:

$$R(A \oplus B) = R(A) \oplus R(B),$$

that can be proved by using induction. Let C be as in section 3.1.3. The reconstruction $R(C)$ is computed recursively, as follows:

$$
\begin{cases}
R(C) = C_1 \text{ for } n = 1, \text{ and} \\
R_1 = C_1 \oplus C_2,\ R_2 = R(R_1),\ R_3 = C_1 \ominus C_7,\ R_4 = R(R_3),\ R_5 = R(C_5),\ R_6 = R_4 \oplus R_5, \\
R_7 = R(C_6),\ R_8 = R_6 \oplus R_7,\ R_9 = R(C_3),\ R_{10} = R(C_4),\ R_{11} = -R_9 \oplus R_{10}, \\
R_{12} = R_4 \oplus R_{11},\ R_{13} = R_6 \oplus R_{10},\ R(C) = [R_2, R_8, R_{12}, R_{13}] \text{ for } n \geq 2.
\end{cases} \tag{15}
$$

It should be noted that the computation of R_1 and R_3 requires $(n^{\log_2 7})/7$ additions/subtractions each because the operation here is comprised of only component additions. On the other hand, the computation $R_2, R_4, R_5, R_7, R_9, R_{10}$ requires $6M^R_{\oplus}(n/2)$ and $5(n/2)^2$ additions for computing $R_6, R_8, R_{11}, R_{12}, R_{13}$, resulting in the following complexities:

$$
\begin{cases}
M^R_{\oplus}(n) \leq & 6M^R_{\oplus}(n/2) + (2/7)n^{\log_2 7} + 5(n/2)^2,\ M^R_{\oplus}(1) = 0, \\
M^R_{\oplus}(n) \leq & 2n^{\log_2 7} + 0.5n^{\log_2 6} - 2.5n^2.
\end{cases} \tag{16}
$$

Example 5. Consider the case of $n = 4$. The length of CM is 49, which is the input for R. Assume that $C = (C_1, \ldots, C_7)$ where $C_i = (P_{7i-6}, \ldots P_{7i})$ for $i = 1, \ldots, 7$. The algorithm in (15) yields:

$R_1 = \quad C_1 \oplus C_2 = [\underbrace{P_1 + P_8}_{r_1}, \underbrace{P_2 + P_9}_{r_2}, \underbrace{P_3 + P_{10}}_{r_3}, \underbrace{P_4 + P_{11}}_{r_4}, \underbrace{P_5 + P_{12}}_{r_5}, \underbrace{P_6 + P_{13}}_{r_6}, \underbrace{P_7 + P_{14}}_{r_7}]$, (7additions)

$R_2 = \quad R(R_1) = [r_1 + r_2, \underbrace{r_8 + r_5}_{r_9} + r_6, \underbrace{r_1 - r_7}_{r_8} - r_3 + r_4, r_9 + r_4]$, (7additions)

$R_3 = \quad C_1 \oplus C_7 = [\underbrace{P_1 + P_{43}}_{r_{10}}, \underbrace{P_2 + P_{44}}_{r_{11}}, \underbrace{P_3 + P_{45}}_{r_{12}}, \underbrace{P_4 + P_{46}}_{r_{13}}, \underbrace{P_5 + P_{47}}_{r_{14}}, \underbrace{P_6 + P_{48}}_{r_{15}}, \underbrace{P_7 + P_{49}}_{r_{16}}]$, (7additions)

$R_4 = \quad R(R_3) = [r_{10} + r_{11}, \underbrace{r_{17} + r_{14}}_{r_{18}} + r_{15}, \underbrace{r_{10} - r_{16}}_{r_{17}} - r_{12} + r_{13}, r_{18} + r_{13}]$, (7additions)

$R_5 = \quad R(C_5) = [P_{29} + P_{30}, \underbrace{r_{19} + P_{33}}_{r_{20}} + P_{34}, \underbrace{P_{29} - P_{35}}_{r_{19}} - P_{31} + P_{32}, r_{20} + P_{32}]$, (7additions)

$R_6 = \quad R_4 \oplus R_5 = [s_1, s_2, s_3, s_4]$, (4 additions)

$R_7 = \quad R(C_6) = [P_{36} + P_{37}, \underbrace{r_{21} + P_{40}}_{r_{22}} + P_{41}, \underbrace{P_{36} - P_{42}}_{r_{21}} - P_{38} + P_{39}, r_{22} + P_{39}]$, (7additions)

$R_8 = \quad R_6 \oplus R_7 = [s_5, s_6, s_7, s_7]$, (4 additions)

$R_9 = \quad R(C_3) = [P_{15} + P_{16}, \underbrace{r_{23} + P_{19}}_{r_{24}} + P_{20}, \underbrace{P_{15} - P_{21}}_{r_{23}} - P_{17} + P_{18}, r_{24} + P_{18}]$, (7additions)

$R_{10} = \quad R(C_4) = [P_{22} + P_{23}, \underbrace{r_{25} + P_{26}}_{r_{26}} + P_{27}, \underbrace{P_{22} - P_{28}}_{r_{25}} - P_{24} + P_{25}, r_{26} + P_{25}]$, (7additions)

$R_{11} = \quad -R_9 \oplus R_{10} = [s_8, s_9, s_{10}, s_{11}]$, (4 additions)

$R_{12} = \quad R_4 \oplus R_{11} = [s_{12}, s_{13}, s_{14}, s_{15}]$, (4 additions)

$R_{13} = \quad R_6 \oplus R_{10} = [s_{16}, s_{17}, s_{18}, s_{19}]$, (4 additions)

$R(A) = \quad [R_2, R_8, R_{12}, R_{13}]$.

The total number of additions is thus 76: one less than in the original case.

The previous and new complexities of CMF, CM and R are summarized in Table 2.

TABLE 2
Complexities of the different sub-operations of algorithms

Method	Operation	Recursion	Non-recursion
Winograd	CMF	$7M_{\oplus}^{CMF}(n/2) + 5(n/2)^2$	$1.33n^{\log_2 7} - 1.33n^2$
	CM	$7M_{\otimes}^{CM}(n/2)$	$n^{\log_2 7}$
	R	$7M_{\oplus}^{R}(n/2) + 8(n/2)^2$	$2.33n^{\log_2 7} - 2.33n^2$
Improved	CMF	$5M_{\oplus}^{CMF}(n/2) + (2/7)n^{\log_2 7} + 2(n/2)^2$	$n^{\log_2 7} + n^{\log_2 5} - 2n^2$
	CM	$7M_{\otimes}^{CM}(\frac{n}{2})$	$n^{\log_2 7}$
	R	$6M_{\oplus}^{R}(n/2) + (2/7)n^{\log_2 7} + 5(n/2)^2$	$2n^{\log_2 7} + 0.5n^{\log_2 6} - 2.5n^2$

From Table 2, the new complexity of WV can be obtained without changing the number of multiplications: the complexity of the new WV requires $Q_1 = CMF_1(A)$, $Q_2 = CMF_2(B)$, $Q_3 = CM(Q_1, Q_2)$, and $Q_4 = R(Q_3)$. It therefore requires

$$2M_{\oplus}^{CMF}(n) + M_{\otimes}^{CM}(n) + M_{\oplus}^{R}(n) = 5n^{\log_2 7} + 0.5n^{\log_2 6} + 2n^{\log_2 5} - 6.5n^2. \quad (17)$$

Employing the complexities listed in Table 2 results in:

$$\begin{cases} M_{\otimes}(n) \leq & n^{2.81}, \\ M_{\oplus}(n) \leq & 4n^{\log_2 7} + 0.5n^{\log_2 6} + 2n^{\log_2 5} - 6.5n^2, \\ M(n) \leq & 5n^{\log_2 7} + 0.5n^{\log_2 6} + 2n^{\log_2 5} - 6.5n^2. \end{cases} \tag{18}$$

5 Stopping the recursion early

In this section, it is shown that the arithmetic complexity of WV can be further improved if the recursion is stopped early, followed by the use of the ordinary algorithm. It should be noted that, in this case, the number of additions is decreased but the number of multiplications is increased. However, the decrease in the number of additions is greater than the increase in the number of multiplications so that the total arithmetic cost is reduced. On the other hand, one should note that this method is useful if decreasing the number of additions is beneficial from a system perspective. For example, the bit addition over binary fields that corresponds to XOR operation in hardware implementations is known to require more space than the bit multiplication that corresponds to AND operation. This method is thus useful for matrix multiplications over binary fields. However, if the multiplication of the matrices is over finite fields with large characteristics or if the entries of the matrices are large numbers, then the multiplication is much more costly than the addition, and increasing the number of multiplications in order to obtain less total arithmetic might not be useful. In such a case, the proposed method described in section 4 offers the best complexity as given in (18). The details of the comparison are included in section 7.

The remainder of this section provides the details for WV complexity. Let $n = m2^k$. WV is assumed to be used k times followed by the use of ordinary multiplication for matrices of size $m \times m$ with the additive complexity of $m^3 - m^2$ and the multiplicative complexity of m^3. The complexity of this method can be computed as follows:

$$\begin{aligned} M(n) &= 7M\left(\tfrac{n}{2}\right) + \tfrac{15}{4}n^2 \\ &= 7^2 M\left(\tfrac{n}{2^2}\right) + \tfrac{15}{4}n^2\left(1 + \tfrac{7}{4}\right) \\ &= 7^3 M\left(\tfrac{n}{2^3}\right) + \tfrac{15}{4}n^2\left(1 + \tfrac{7}{4} + \left(\tfrac{7}{4}\right)^2\right) \\ &\vdots \\ &= 7^k M\left(\tfrac{n}{2^k}\right) + \tfrac{15}{4}n^2\left(1 + \tfrac{7}{4} + \left(\tfrac{7}{4}\right)^2 + \ldots + \left(\tfrac{7}{4}\right)^{k-1}\right). \end{aligned}$$

So this results in

$$M(n) = 7^k\left(M\left(\frac{n}{2^k}\right) + \frac{5n^2}{4^k}\right) - 5n^2 = 7^k[M(m) + 5m^2] - 5n^2 = 2(m+2)m^2 7^k - 5(m2^k)^2. \tag{19}$$

The addition and multiplication complexities are obtained separately as follows:

$$\begin{cases} M_{\otimes}(n) = m^3 7^k, \\ M_{\oplus}(n) = (m+4)m^2 7^k - 5(m2^k)^2. \end{cases} \tag{20}$$

The complexities of WV for different cut-off values m are presented in Table 3. It can be concluded that the best arithmetic complexity of WV matrix multiplication is obtained when the recursion is stopped at the point when the size of the matrices becomes eight. In this case, the result is

$$\begin{cases} M_{\otimes}(n) \leq & 1.49n^{2.81}, \\ M_{\oplus}(n) \leq & 2.26n^{2.81} - 5n^2, \\ M(n) \leq & 3.73n^{2.81} - 5n^2. \end{cases} \tag{21}$$

TABLE 3
Complexities obtained by stopping the recursion early for Winograd's variant algorithm

m	Addition	Multiplication	Total
1	$5n^{2.81} - 5n^2$	$n^{2.81}$	$6n^{2.81} - 5n^2$
2	$3.43n^{2.81} - 5n^2$	$1.14n^{2.81}$	$4.53n^{2.81} - 5n^2$
4	$2.61n^{2.81} - 5n^2$	$1.61n^{2.81}$	$4.57n^{2.81} - 5n^2$
8	$2.23n^{2.81} - 5n^2$	$1.49n^{2.81}$	$3.73n^{2.81} - 5n^2$
16	$2.13n^{2.81} - 5n^2$	$1.71n^{2.81}$	$3.83n^{2.81} - 5n^2$
32	$2.19n^{2.81} - 5n^2$	$1.95n^{2.81}$	$4.14n^{2.81} - 5n^2$

On the other hand, when this approach is applied to the proposed method, the result is not so beneficial because the addition of the CMFs used in the proposed method reduces the amount of improvements. For example, if the recursion is stopped when $n = 2$, then it is found that $CMF(n) \cong 1.14n^{2.81} - 1.33n^{2.32} - 0.67n$ which is grater than in (14). However, the total arithmetic complexity is improved to $(3.55n^{2.81} + 0.148n^{2.59} + 1.02n^{2.32} - 6.5n^2)$, which is the best-known complexity, as explained in the following sections.

6 Further improvements using block recombination method

This section describes the use of the results from section 4 together with a block recombination method [6] in order to improve the arithmetic cost of matrix multiplications to $(3.55n^{2.81} + 0.148n^{2.59} + 1.02n^{2.32} - 6.5n^2)$. The following is the main idea: Let A, B, C, D be matrices of dimensions n. The method is based on the observation of the linearity of the reconstruction step of the block recombination method, i.e., based on the following equation:

$$R(C_1) \oplus R(C_2) = R(C_1 + C_2). \tag{22}$$

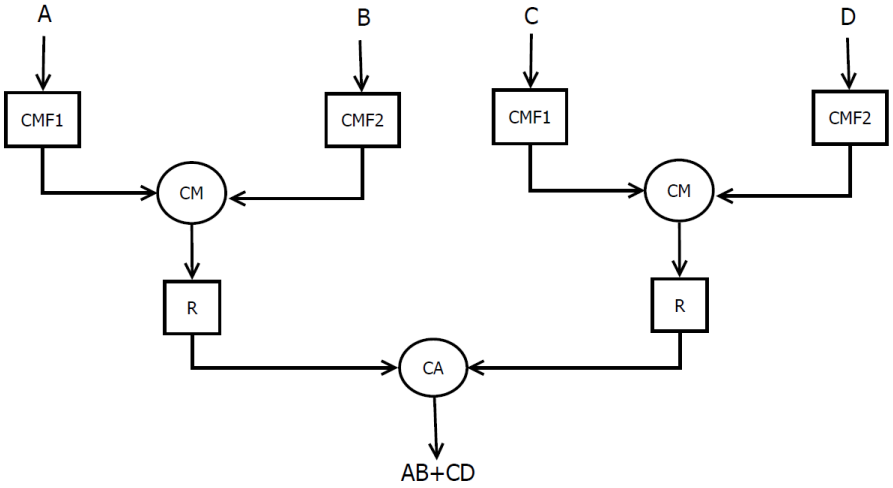
This equation provides improvements for the computation of $AB + CD$ where A, B, C and D are $n \times n$ matrices. It should be noted that the direct computation of $AB + CD$ requires $Q_1 = CMF_1(A)$; $Q_2 = CMF_2(B)$; $Q_3 = CMF_1(C)$; $Q_4 = CMF_2(D)$; $Q_5 = CM(Q_1, Q_2)$; $Q_6 = CM(Q_3, Q_4)$; $Q_7 = R(Q_5)$;

$Q_8 = R(Q_6)$; and finally due to the final addition of AB and CD, n^2 additions. The total cost of the arithmetic complexity of computing $AB + CD$ is thus

$$4M_{\oplus}^{CMF}(n) + 2M_{\otimes}^{CM}(n) + 2M_{\oplus}^{R}(n) + n^2.$$

The results are therefore $(14n^{2.81} - 11n^2)$ with the use of SA, $(12n^{2.81} - 9n^2)$ with the use of WV, and $(10n^{2.81} + n^{2.59} + 4n^{2.32} - 12n^2)$ with the use of the improved algorithm presented in section 4. Figure 2 shows the computation.

Fig. 2. Data flow of block recombination for direct computation of $AB + CD$

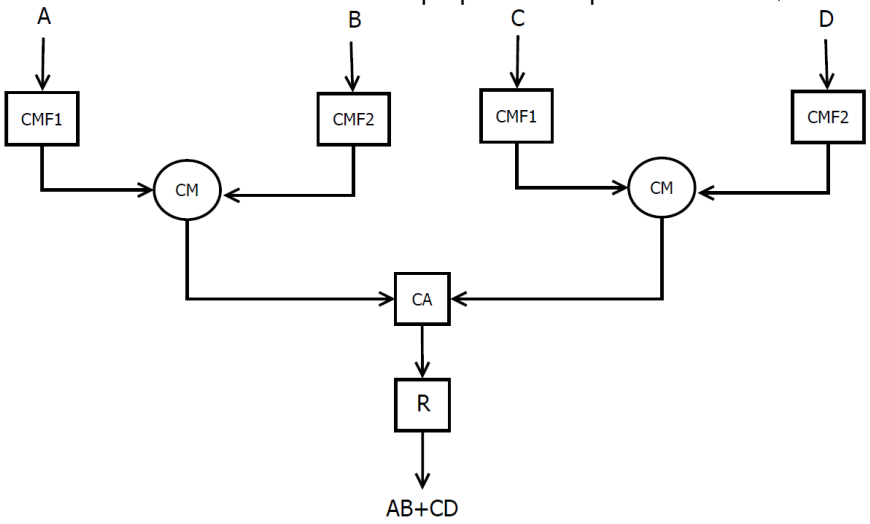


On the other hand, block recombination method combined with (22) requires $Q_1 = CMF_1(A)$; $Q_2 = CMF_2(B)$; $Q_3 = CMF_1(C)$; $Q_4 = CMF_2(D)$ $Q_5 = CM(Q_1, Q_2)$; $Q_6 = CM(Q_3, Q_4)$; $Q_7 = CA(Q_5, Q_6)$; and $Q_8 = R(Q_7)$. The total cost of the arithmetic complexity of computing $AB + CD$ is thus

$$4M_{\oplus}^{CMF}(n) + 2M_{\otimes}^{CM}(n) + M_{\oplus}^{R}(n) + n^{2.81},$$

where $n^{2.81}$ is the complexity of $CA(Q_5, Q_6)$. This process is illustrated in Figure 3. When the complexities given in Table 1 and Table 2 are used, the resulting total arithmetic cost of computing $AB + CD$ is $(12.33n^{2.81} - 9.33n^2)$ with the use of SA, $(10.67n^{2.81} - 7.67n^2)$ with the use of WV, and $(9n^{2.81} + 0.5n^{2.59} + 4n^{2.32} - 10.5n^2)$ with the use of the improved algorithm presented in section 4.

Fig. 3. Data flow of the block recombination for the proposed computation of $AB + CD$



A similar method can then provide further improvement in the complexities if the computation starts with ordinary multiplication. Let A and B be two matrices of dimensions n, and let $C = AB$. The computation starts with the following:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}.$$

The first step is to compute the CMF operation applied to A_{ij}'s and B_{ij}'s for $1 \leq i, j \leq 2$. The cost of this step is eight CMFs applied to $n/2 \times n/2$ size matrices so that the size of each CMF is $n^{\log_2 7}/7$. Eight component multiplications are then performed: $A_{11}B_{11}$, $A_{12}B_{21}$, $A_{11}B_{12}$, $A_{12}B_{22}$, $A_{21}B_{11}$, $A_{22}B_{21}$, $A_{21}B_{12}$, and $A_{22}B_{22}$. For this step, CM is applied to vectors of sizes $n^{\log_2 7}/7$. Then, rather than applying operation R to these products, the component additions (CA) are computed: $A_{11}B_{11} + A_{12}B_{21}$, $A_{11}B_{12} + A_{12}B_{22}$, $A_{21}B_{11} + A_{22}B_{21}$, and $A_{21}B_{12} + A_{22}B_{22}$. It should be noted that four CAs with sizes $n^{\log_2 7}/7$ are required. The final step is to apply operation R to those four sums of the products, for which four R operations are required. Figure 4 illustrates how the method works. The result is a total arithmetic cost of

$$8M_{\oplus}^{CMF}(n/2) + 8M_{\otimes}^{CM}(n/2) + 4M_{\oplus}^{CA}(n/2) + 4M_{\oplus}^{R}(n/2).$$

With the use of Table 2, the new improved complexities can be obtained as follows:

$$M(n) \leq 4n^{2.81} + 1.6n^{2.32} - 6.5n^2 + 0.3n^{2.59}.$$

It should be noted that this bound is superior to the bound of the original WV obtained by stopping the algorithm when the dimension of matrices is two that is $4.53n^{2.81} - 5n^2$.

Moreover, if the matrices are initially divided into four parts and the ordinary multiplication is used, then we need

$$32M_{\oplus}^{CMF}(n/4) + 64M_{\otimes}^{CM}(n/4) + 48M_{\oplus}^{CA}(n/4) + 16M_{\oplus}^{R}(n/4)$$

This calculation results in

$$M(n) \leq 3.59n^{2.81} + 1.28n^{2.32} - 6.5n^2 + 0.22n^{2.59}.$$

More generally, if $2^i \times 2^i$ dimensional matrices are formed initially and ordinary multiplication is used, then the total arithmetic cost is

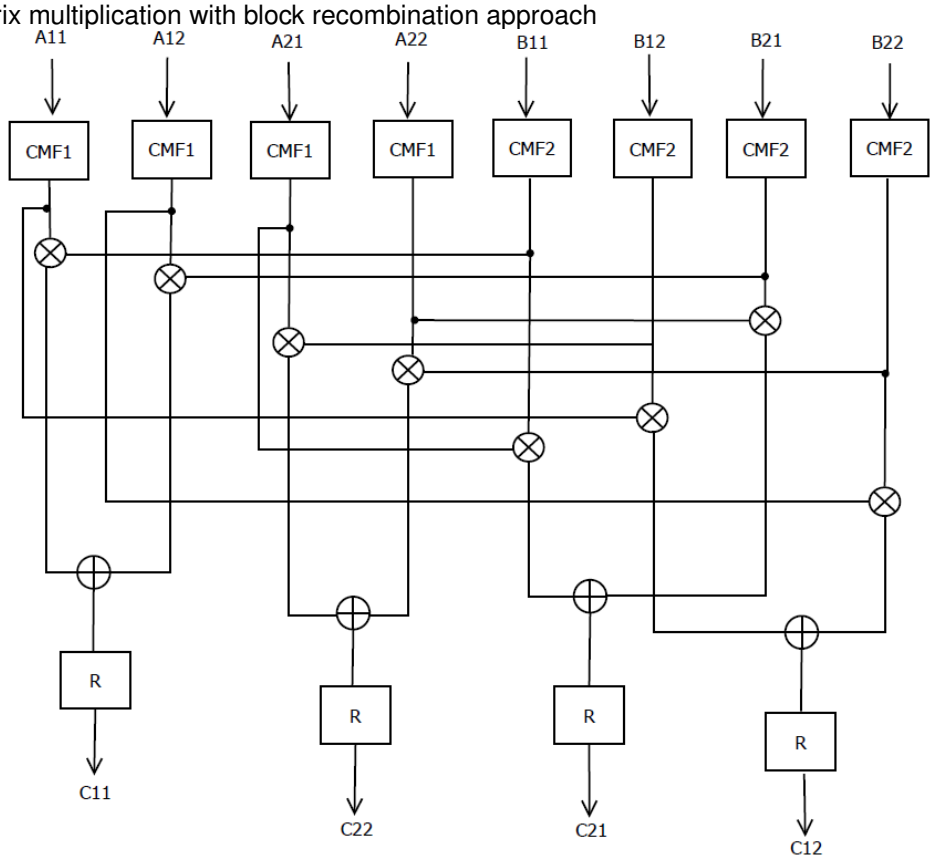
$$2 \cdot 2^{2i} M_{\oplus}^{CMF}(n/2^i) + 2^{3i} M_{\otimes}^{CM}(n/2^i) + (2^{3i} - 2^{2i}) M_{\oplus}^{CA}(n/2^i) + 2^{2i} M_{\oplus}^{R}(n/2^i).$$

For $i = 3$, the following complexities are obtained:

$$\begin{cases} M_{\otimes}(n) \leq & 1.49n^{2.81}, \\ M_{\oplus}(n) \leq & 2.06n^{2.81} + 1.02n^{2.32} - 6.5n^2 + 0.148n^{2.59}, \\ M(n) \leq & 3.55n^{2.81} + 1.02n^{2.32} - 6.5n^2 + 0.148n^{2.59}, \end{cases} \tag{23}$$

which, to the best of our knowledge, represents an improvement over the best-known arithmetic cost reported in the literature.

Fig. 4. Matrix multiplication with block recombination approach



7 COMPLEXITY COMPARISON

This section provides the complexity results for the cases. We recall that this study deals with the arithmetic complexity and that the total number of operations in the ring over which the matrices are defined are counted. If it is assumed that the cost of the ring operations $+, -$, and $*$ are almost equal, then the best complexities should be compared: $3.73n^{2.81} + O(n^2)$ for WV and $3.55n^{2.81}n^2 + O(n^\alpha)$, $\alpha \leq 2.59$ for the proposed algorithm. Possible examples of this case include Boolean matrix multiplications and matrix multiplications over $\mathbb{F}_p$ in which $\lceil \log_2 p \rceil$ is less than the machine word size. Care should be taken, however, with matrices that have entries stored in more than one word. Such large numbers are used in cryptographic applications, in which case, the cost of multiplication is generally greater than the cost of addition. It should be recalled that although stopping the recursion when $n = 8$ and then using ordinary multiplication yields the best arithmetic complexity, the number of multiplications also increases from $n^{2.81}$ to $1.49n^{2.81}$. The $6n^{2.81} + O(n^2)$ complexity should therefore be used for WV and the $5n^{2.81}n^2 + O(n^\alpha)$ complexity for the proposed algorithm because they include less multiplications than the others. As verification, the previous and new complexities have been compared for matrices over binary fields, for matrices with entries whose lengths are fitted to the word size of processor, and for matrices with entries whose lengths are greater than the word size of processor.

TABLE 4
Total arithmetic complexity comparison

WV	Proposed	n	WV	Proposed	Improvement %
$3.73n^{2.81} - 5n^2$	$3.55n^{2.81} + 0.15n^{2.59}$ $+1.02n^{2.32} - 6.5n^2$	256	21175027	20686080	2.31
		4096	51544115180	49598704883	3.77
		16384	2.5x10^{12}	2.4x10^{12}	4.14
		262,144	6.07x10^{15}	5.80x10^{15}	4.52

The first comparison involves the comparison of product of matrices over binary fields which are widely used in cryptographic applications [7], [17]. The cost of addition and multiplication of bits can be assumed to be identical in software implementation for which the total of arithmetic cost can be compared. That is, $3.73n^{2.81} + O(n^2)$ can be compared with the $3.55n^{2.81} + O(n^\alpha)$ where $\alpha \leq 2.58$. As can be seen from Table 4, the improvements in this case are between 2.31% and 4.52%. On the other hand, for hardware implementations of matrix multiplication over binary fields, the weights of the additions and the multiplications should be considered separately. For some platforms the addition, or the XOR gate, is more costly than the multiplication, or the AND gate. For example, an XOR gate requires twice as many transistors as an AND gate for hardware implementations using ASIC (Application Specific Integrated Circuit) technology. Therefore, we should compare $2M_\oplus(n) + M_\otimes(n)$ to measure the space complexity. In this case the proposed algorithm with $2.06n^{2.81} + O(n^\alpha)$ additions and $1.49n^{2.81}$ multiplications results

in improvements between 2.85% and 6.09% over the WV with $2.26n^{2.81} + O(n^2)$ additions and $1.49n^{2.81}$ multiplications for $128 \leq n \leq 65536$. The results are tabulated in Table 5.

TABLE 5
Comparison of the $2M_{\oplus}(n) + M_{\otimes}(n)$ space complexity for hardware implementation over binary fields

WV	Proposed	n	WV	Proposed	Improvement %
$2M_{\oplus}(n) + M_{\otimes}(n) = 6.01n^{2.81} - 10n^2$	$2M_{\oplus}(n) + M_{\otimes}(n) = 5.61n^{2.81} + 0.30n^{2.59} + 2.04n^{2.32} - 13n^2$	128	4785653	4649320	2.85
		256	33991094	32782606	3.55
		512	239903738	229943236	4.15
		4096	83018363918	78573891836	5.35
		65536	2x10^{14}	1.87x10^{14}	6.09

In the case of the multiplication of matrices with entries whose sizes are less than word size of processor, the results are similar to those for matrices over binary fields because one may assumed that most of the processors perform additions and multiplications of those numbers in an approximately equal time. It can thus be stated that the improvements are about 4% as indicated in Table 4.

The final step is to analyze the multiplication of matrices with entries whose lengths are greater than the word size of processor. In this case, we know that the number of multiplications needed for such numbers is much larger than the number of additions. More precisely, if the number of words required for storing the entries of matrices is ℓ, ℓ^2 multiplications and $(\ell-1)^2$ additions of words are needed in order to multiply the entries of the matrices using the school-book method. On the other hand, ℓ additions of words are required for adding two entries. It should be noted that the school-book method for multiplication is efficient only for a small value ℓ. More efficient algorithms, such as Karatsuba multiplication, are available for larger ℓ values. However, for this study, the complexities for $\ell < 6$ were analyzed, and the school-book method was used. The arithmetic complexity is then computed as follows: Let the multiplication and the addition complexities of WV be $M_{\otimes}^W$ and $M_{\oplus}^W$, respectively and the multiplication and the addition complexities of the proposed algorithm be $M_{\otimes}^P$ and $M_{\oplus}^P$, respectively. Then, $(\ell^2 + (\ell-1)^2)M_{\otimes}^W + \ell M_{\oplus}^W$ is compared with $(\ell^2 + (\ell-1)^2)M_{\otimes}^P + \ell M_{\oplus}^P$. It should be noted that in (21), the complexity of WV yields better results for $\ell = 2, 3$ and that, in (5), the complexity of WV yields better results for $\ell > 3$. On the other hand, in (23), the complexity of the proposed algorithm produces better results for $\ell = 2$ and in (18), the complexity of the proposed algorithm gives better results for $\ell \geq 3$ because a smaller number of multiplications are used in the latter case than in the former. The computations show that, for $n > 4096$, the improvements are about 4%, 8% and 7% for $\ell = 3$, 4, 5 respectively.

8 Conclusion

We have improved the arithmetic complexity of Strassen-like matrix multiplication from $(6n^{2.81} - 5n^2)$ to $(5n^{2.81} + 0.5n^{2.59} + 2n^{2.32} - 6.5n^2)$ for $n = 2^k$ and from $(3.73n^{2.81} - 5n^2)$ to $(3.55n^{2.81} + 0.148n^{2.59} +$

$1.02n^{2.32} - 6.5n^2$) for $n = 8 \cdot 2^k$. These results correspond to improvements between 2% and 8% depending on the size of the entries of the matrices and the implementation platform.

REFERENCES

[1] D. Bini, M. Capovani, F. Romani, and G. Lotti. $O(n^{2.7799})$ complexity for $n*n$ approximate matrix multiplication. *Inf. Process. Lett.*, 8(5):234–235, 1979.

[2] H. Cohn, R. D. Kleinberg, B. Szegedy, and C. Umans. Group-theoretic algorithms for matrix multiplication. In *FOCS*, pages 379–388, 2005.

[3] H. Cohn and C. Umans. A group-theoretic approach to fast matrix multiplication. In *FOCS*, pages 438–449, 2003.

[4] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[6] M. A. Hasan, N. Meloni, A. H. Namin, and C. Nègre. Block recombination approach for subquadratic space complexity binary field multiplication based on toeplitz matrix-vector product. *IEEE Trans. Computers*, 61(2):151–163, 2012.

[7] A. Joux. *Algorithmic cryptanalysis*. Chapman & Hall/CRC, 1st edition, 2009.

[8] I. Kaporin. The aggregation and cancellation techniques as a practical tool for faster matrix multiplication. *Theor. Comput. Sci.*, 315(2-3):469–510, 2004.

[9] V. Y. Pan. Strassen's algorithm is not optimal: Trililnear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations. In *FOCS*, pages 166–176, 1978.

[10] V. Y. Pan. New fast algorithms for matrix operations. *SIAM J. Comput.*, 9(2):321–342, 1980.

[11] V. Y. Pan. *How to Multiply Matrices Faster*, volume 179 of *Lecture Notes in Computer Science*. Springer, 1984.

[12] R. L. Probert. On the additive complexity of matrix multiplication. *SIAM J. Comput.*, 5(2), 1976.

[13] A. Schönhage. Partial and total matrix multiplication. *SIAM J. Comput.*, 10(3):434–455, 1981.

[14] A. Stothers. *On the complexity of matrix multiplication*. PhD thesis, University of Edinburgh, 2010.

[15] V. Strassen. Gaussian Elimination is not Optimal. *Numer. Math.*, 13:354–356, 1969.

[16] V. Strassen. The asymptotic spectrum of tensors and the exponent of matrix multiplication. In *FOCS*, pages 49–54, 1986.

[17] B. Gregory V. *Algebraic Cryptanalysis (1. ed.)*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[18] V. V. Williams. Multiplying matrices faster than cppersmith-winograd. In *STOC*, pages 887–898, 2012.

[19] S. Winograd. On multiplication of 2x2 matrices. *Linear Algebra and Application*, 4:381 – 388, 1971.